The Complexity of Coverage*

Krishnendu Chatterjee[§] Luca de Alfaro[§] Rupak Majumdar[†]

[§] CE, University of California, Santa Cruz, USA

[†] CS, University of California, Los Angeles, USA

{c_krish,luca}@soe.ucsc.edu, rupak@cs.ucla.edu

April 2008

Technical Report No. UCSC-SOE-08-03

School of Engineering, University of California, Santa Cruz, CA, USA

Abstract

We study the problem of generating a test sequence that achieves maximal coverage for a reactive system under test. We formulate the problem as a repeated game between the tester and the system, where the system state space is partitioned according to some coverage criterion and the objective of the tester is to maximize the set of partitions (or coverage goals) visited during the game. We show the complexity of the maximal coverage problem for non-deterministic systems is PSPACE-complete, but is NP-complete for deterministic systems. For the special case of non-deterministic systems with a re-initializing "reset" action, which represent running a new test input on a re-initialized system, we show that the complexity is again co-NP-complete. Our proof technique for reset games uses randomized testing strategies that circumvent the exponentially large memory requirement in the deterministic case.

1 Introduction

Code coverage is a common metric in software and hardware testing that measures the degree to which an implementation has been tested with respect to some criterion. In its simplest form, one starts with a model of the program, and a partition of the behaviors of the model into *coverage goals* [3]. A *test* is a sequence of inputs that determines a behavior of the program. The aim of testing is to explore as many coverage goals as possible, ideally as quickly as possible. In this paper, we give complexity results for several coverage problems. The problems are very basic in nature: they consist in deciding whether a certain level of coverage can be attained in a given system. It is thus somewhat surprising that the problems have not been considered previously in the literature.

Finite-state directed graphs have been used as program models for test generation of reactive systems for a long time (see [15, 6] for surveys). A coverage goal is a partition of the states of the graph, and a test is a sequence of labels that determine a path in the graph. The maximal coverage test generation problem is to hit as many partitions as possible using a minimum number of tests. In the special case the partitions coincide with the states, the maximal coverage problem reduces to the Chinese postman problem for which there are efficient (polynomial time) algorithms [7]. In this

*This research was suppored in part by the NSF grants CCR-0132780 and CNS-0720884.

paper, we show that the maximal coverage problem becomes NP-complete for graphs with general partitions. We also distinguish between *system complexity* (the complexity of the problem in terms of the size of the graph) and the *coverage complexity* (the complexity of the problem in terms of the number of coverage goals). Then, the problem is NLOGSPACE in the size of the graph (but that algorithm uses space polynomial in the number of propositions).

We consider the special case where the graph has a special "reset" action that takes it back to the initial state. This corresponds in a testing setting to the case where the system can be re-initialized before running a test. In this case, the maximal coverage problem remains polynomial, even with general partitions.

Directed graphs form a convenient representation for deterministic systems, in which all the choices are under the control of the tester. Testing of non-deterministic systems in which certain actions are controllable (under the control of the tester) and other actions are uncontrollable lead to *game graphs* [14]. A game graph is a directed labeled graph where the nodes are partitioned into tester-nodes and system-nodes, and while the tester can choose the next input at a tester node, the system non-deterministically chooses the next state at a system node. Then, the test generation problem is to generate a test set that achieves maximal coverage no matter how the system moves. For general game graphs, we show the complexity of the maximal coverage problem is PSPACE-complete. However, there is an algorithm that runs in time linear in the size of the game graph but exponential in the number of coverage goals. Again, the re-initializability assumption reduces the complexity of coverage: in case there is a re-initialization strategy of the tester from any system state, the maximal coverage problem for games is co-NP-complete. Dually, we show that the problem of whether it is possible to win a safety game while visiting fewer than a specified number of partitions is NP-complete.

Finally, we consider the coverage problem in bounded time, consisting in checking whether a specified number of partitions can be visited in a pre-established number of steps. We show that the problem is NP-complete for graphs, and is PSPACE-complete for game graphs.

Optimization problems arising out of test generation have been studied before in the context of both graphs and games [1, 10, 14, 5]. However, to the best of our knowledge, the complexities of the coverage problems studied here have escaped attention so far.

While we develop our theory for the finite-state, discrete case, we can derive similar results for more general models, such as those incorporating incomplete information (the tester can only observe part of the system state) or timing. For timed systems modeled as timed automata, the maximal coverage problem is PSPACE-complete. For timed games as well as for (finite state) game graphs with incomplete information, the maximal coverage problem becomes EXPTIME-complete.

2 Definitions

In this section we define *labeled graphs* and *labeled games*, and then define the two decision problems of coverage, namely, *maximal coverage* problem and *coverage with bounded time* problem. We start with definition of graphs and games.

Definition 1 (Labeled graphs) *A* labeled graph $\mathcal{G} = ((V, E), v_{in}, \mathsf{AP}, \mathcal{L})$ *consists of the following component:*

1. *A finite directed graph with vertex set V and edge set E;*

2. *the initial vertex* v_{in}*;*

3. *a finite set of atomic propositions* AP*;*

4. *a labeling function* $\mathcal{L}$ *that assigns to each vertex* v *the set* $\mathcal{L}(v)$ *of atomic propositions true at* s*.*

For technical convenience we will assume that for all vertices $v \in V$*, there exists* $u \in V$ *such that* $(v, u) \in E$*, i.e., each vertex has at least one out-going edge.*

Paths in graphs and reachability. Given a labeled graph $\mathcal{G}$, a *path* ω in $\mathcal{G}$ is a infinite sequence of vertices $\langle v_0, v_1, v_2 \ldots\rangle$ starting from the initial vertex v_{in} (i.e., $v_0 = v_{in}$) such that for all $i \geq 0$ we have $(v_i, v_{i+1}) \in E$. A vertex v_i is reachable from v_{in} if there is a path $\omega = \langle v_0, v_1, v_2 \ldots\rangle$ in $\mathcal{G}$ and $j \geq 0$ such that the vertex v_j in ω is the vertex v_i.

Definition 2 (Labeled game graphs) *A* labeled game graph $\mathcal{G} = ((V, E), (V_1, V_2), v_{in}, \mathsf{AP}, \mathcal{L})$ *consists of the components of a labeled graph along with a partition of the finite vertex set* V *into* (V_1, V_2)*. The vertices in* V_1 *are player 1 vertices where player 1 chooses outgoing edges, and analogously, the vertices in* V_2 *are player 2 vertices where player 2 chooses outgoing edges. Again for technical convenience we will assume that for all vertices* $v \in V$*, there exists* $u \in V$ *such that* $(v, u) \in E$*, i.e., each vertex has at least one out-going edge.*

Plays and strategies in games. A *play* in a game graph is a path in the underlying graph of the game. A strategy for a player in a game is a recipe to specify how to extend the prefix of a play. Formally, a strategy π_1 for player 1 is a function $\pi_1 : V^* \cdot V_1 \to V$ that takes a finite sequence of vertices $w \cdot v$ ending in a player 1 vertex v, where $w \in V^*$ and $v \in V_1$, representing the history of the play so far, and specifies the next vertex $\pi_1(w \cdot v)$ choosing an out-going edge (i.e., $(v, \pi_1(w \cdot v)) \in E$. A strategy $\pi_2 : V^* \cdot V_2 \to V$ is defined analogously. We denote by Π_1 and Π_2 the set of all strategies for player 1 and player 2, respectively. Given strategies π_1 and π_2 for player 1 and player 2, there is a unique play (or a path) $\omega(v_{in}, \pi_1, \pi_2) = \langle v_0, v_1, v_2, \ldots$ such that (a) $v_0 = v_{in}$; (b) for all $i \geq 0$, if $v_i \in V_1$, then $\pi_1(v_0 \cdot v_1 \ldots \cdot v_i) = v_{i+1}$; and if $v_i \in V_2$, then $\pi_2(v_0 \cdot v_1 \ldots \cdot v_i) = v_{i+1}$.

Controllably recurrent graphs and games. Along with general labeled graphs and games, we will also consider graphs and games that are *controllably recurrent*. A labeled graph $\mathcal{G}$ is *controllably recurrent* if for every vertex v_i that is reachable from v_{in}, there is a path starting from v_i that reaches v_{in}. A labeled game graph $\mathcal{G}$ is *controllably recurrent* if for every vertex v_i that is reachable from v_{in} in the underlying graph, there is a strategy π_1 for player 1 such that against all player 2 strategies π_2, the path starting from v_i given the strategies π_1 and π_2 reaches v_{in}. Controllable recurrence models the natural requirement that systems under test are *re-initializable*, that is, from any reachable state of the system, there is always a way to bring the system back to its initial state no matter how the system behaves.

The maximal coverage problem. The *maximal coverage problem* asks whether at least m different propositions can be visited. We now define the problem formally for graphs and games. Given a path $\omega = \langle v_0, v_1, v_2, \ldots\rangle$, let $\mathcal{L}(\omega) = \bigcup_{i \geq 0} \mathcal{L}(v_i)$ be the set of propositions that appear in ω. Given a labeled graph $\mathcal{G}$ and $0 \leq m \leq |\mathsf{AP}|$, the maximal coverage problem asks whether there is path ω such that $|\mathcal{L}(\omega)| \geq m$. Given a labeled game graph $\mathcal{G}$ and $0 \leq m \leq |\mathsf{AP}|$, the maximal

coverage problem asks whether player 1 can ensure that at least m propositions are visited, i.e., whether

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2))| \geq m.$$

It may be noted that $\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2))| \geq m$ iff there exists a player 1 strategy π_1^* such that for all player 2 strategies π_2^* we have $|\mathcal{L}(\omega(v_{in}, \pi_1^*, \pi_2^*))| \geq m$.

The maximal *state coverage* problem is the special case of the maximal coverage problem where $\mathsf{AP} = V$ and for each $v \in V$ we have $\mathcal{L}(v) = \{v\}$. That is, each state has its own label, and there are $|V|$ singleton partitions.

The coverage with bounded time problem. The *coverage with bounded time problem* asks whether at least m different propositions can be visited within k-steps. We now define the problem formally for graphs and games. Given a path $\omega = \langle v_0, v_1, v_2, \ldots \rangle$ and $k \geq 0$, we denote by $\omega \restriction k$ the prefix of the path of length $k+1$, i.e., $\omega \restriction k = \langle v_0, v_1, \ldots, v_k \rangle$. Given a path $\omega = \langle v_0, v_1, v_2, \ldots \rangle$ and $k \geq 0$, we denote by $\mathcal{L}(\omega \restriction k) = \bigcup_{0 \leq i \leq k} \mathcal{L}(v_i)$. Given a labeled graph $\mathcal{G}$ and $0 \leq m \leq |\mathsf{AP}|$ and $k \geq 0$, the coverage with bounded time problem asks whether there is path ω such that $|\mathcal{L}(\omega \restriction k)| \geq m$. Given a labeled game graph $\mathcal{G}$ and $0 \leq m \leq |\mathsf{AP}|$, the maximal coverage problem asks whether player 1 can ensure that at least m propositions are visited within k-steps, i.e., whether

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2) \restriction k)| \geq m.$$

It may be noted that $\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2) \restriction k)| \geq m$ iff there exists a player 1 strategy π_1^* such that for all player 2 strategies π_2^* we have $|\mathcal{L}(\omega(v_{in}, \pi_1^*, \pi_2^*) \restriction k)| \geq m$.

System-tester game. A *system* $\mathcal{S} = (Q, \Sigma, q_{in}, \Delta, \mathsf{AP}, \mathcal{L})$ consists of the following components:

- A finite set Q of states with the starting state q_{in}.
- A finite alphabet Σ of input letters.
- A transition relation $\Delta \subseteq Q \times \Sigma \times Q$.
- A finite set of atomic propositions AP and a labeling function $\mathcal{L}$ that assigns to each state q the set of atomic propositions true at q.

We consider *total* systems such that for all $q \in Q$ and $\sigma \in \Sigma$, there exists $q' \in Q$ such that $(q, \sigma, q') \in \Delta$. A system is *deterministic* if for all $q \in Q$ and $\sigma \in \Sigma$, there exists exactly one q' such that $(q, \sigma, q') \in \Delta$. The tester selects an input letter at every stage and the system resolves the non-determinism in transition to choose the successor state. The goal of the tester is to visit as many different propositions as possible. The interaction between the system and the tester can be reduced to a labeled game graph $\mathcal{G} = ((V, E), (V_1, V_2), v_{in}, \mathsf{AP}, \mathcal{L}')$ as follows:

- *Vertices and partition.* $V = Q \cup Q \times \Sigma$; $V_1 = Q$ and $V_2 = Q \times \Sigma$; and $v_{in} = q_{in}$.
- *Edges.* $E = \{(q, (q, \sigma)) \mid q \in Q, \sigma \in \Sigma\} \cup \{((q, \sigma), q') \mid (q, \sigma, q') \in \Delta\}$.
- *Labeling.* $\mathcal{L}'(q) = \mathcal{L}(q)$ and $\mathcal{L}'((q, \sigma)) = \mathcal{L}'(q)$.

The coverage question for game between tester and system can be answered by answering the question in the game graph. Also observe that if the system is deterministic, then for all player 2 vertices in the game graph, there is exactly one out-going edge, and hence the game can be reduced to a labeled graph. In this paper we will present all the results for the labeled graph and game model. All the upper bounds we provide follow also for the game between tester and system. All the lower bounds we present can also be easily adapted to the model of the game between system and tester.

3 The Complexity of Maximal Coverage Problems

In this section we study the complexity of the maximal coverage problem. In subsection 3.1 we study the complexity for graphs, and in subsection 3.2 we study the complexity for game graphs.

3.1 Graphs

We first show that the maximal coverage problem for labeled graphs is NP-complete.

Theorem 1 *The maximal coverage problem for labeled graphs is NP-complete.*

Proof. The proof consists of two parts. We present them below.

1. *In NP.* The maximal coverage problem is in NP can be proved as follows. Given a labeled game graph $\mathcal{G}$, let $n = |V|$. We show first that if there is a path ω in $\mathcal{G}$ such that $|\mathcal{L}(\omega)| \geq m$, then there is a path ω' in $\mathcal{G}$ such that $|\mathcal{L}(\omega' \upharpoonright m \cdot n)| \geq m$. If ω visits at least m propositions, and there is a cycle in ω that does not visit a new proposition that is already visited in the prefix, then the cycle segment can be removed from ω and still the resulting path visits m propositions. Hence if the answer to the maximal coverage problem is "Yes", then there is a path ω' of length at most $m \cdot n$ that is a witness to the "Yes" answer. Since $m \leq |\mathsf{AP}|$, it follows that the problem is in NP.

2. *NP-hardness.* Now we show that the maximal coverage problem is NP-hard, and we present a reduction from the SAT-problem. Consider a SAT formula Φ, and let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of variables and $C_1, C_2, \ldots, C_m$ be the set of clauses. For a variable $x_j \in X$, let

   (a) $\textsc{t}(x_j) = \{\ell \mid x_j \in C_\ell\}$ be the set of indices of the set of clauses C_ℓ that is satisfied if x_j is set to be true; and

   (b) $\textsc{f}(x_j) = \{\ell \mid \overline{x}_j \in C_\ell\}$ be the set of indices of the set of clauses C_ℓ that is satisfied if x_j is set to be false.

   Without loss of generality, we assume that $\textsc{t}(x_j)$ and $\textsc{f}(x_j)$ are non-empty for all $1 \leq j \leq n$ (this is because, for example, if $\textsc{f}(x_j) = \emptyset$, then we can set x_j to be true and reduce the problem where the variable x_j is not present). For a finite set $F \subseteq \mathbb{N}$ of natural numbers, let $\max(F)$ and $\min(F)$ denote the maximum and minimum number of F, respectively, for an element $f \in F$ that is not the maximal element let $\mathsf{next}(f, F)$ denote the next highest element to f that belongs to F; i.e., (a) $\mathsf{next}(f, F) \in F$; (b) $f < \mathsf{next}(f, F)$; and (c) if $j \in F$ and $f < j$, then $\mathsf{next}(f, F) \leq j$. We construct a labeled game graph $\mathcal{G}^\Phi$ as follows. We first present an intuitive description: there are states labeled $x_1, x_2, \ldots, x_n, x_{n+1}$, and all of them are labeled

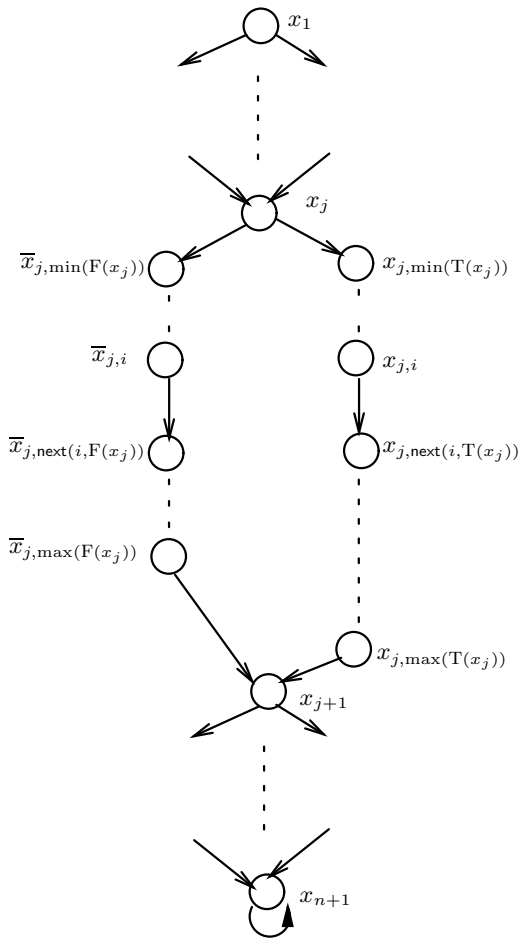

Figure 1: The NP-hardness reduction in picture.

by a single proposition. The state x_{n+1} is an absorbing state (state with a self-loop only), and all other x_i state has two successors. The starting is x_1. In every state x_i given the right choice we visit in a line a set of states that are labeled by clauses that are true if x_i is true; and given the left choice we visit in a line a set of states that are labeled by clauses that are true if x_i is false; and then we move to state x_{i+1}. We now formally describe every component of the labeled graph $\mathcal{G}^\Phi = \langle (V^\Phi, E^\Phi), v_{in}^\Phi, \mathsf{AP}^\Phi, \mathcal{L}^\Phi \rangle$.

(a) The set of vertices is

$$\begin{aligned} V^\Phi &= \{x_i \mid 1 \le i \le n+1\} \\ &\cup \{x_{j,i} \mid 1 \le j \le n, i \in \mathrm{T}(x_j)\} \cup \{\overline{x}_{j,i} \mid 1 \le j \le n, i \in \mathrm{F}(x_j)\}. \end{aligned}$$

There is a vertex for every variable, and a vertex x_{n+1}. There is a vertex $x_{j,i}$ iff $C_i \in \mathrm{T}(x_j)$, and there is a vertex $\overline{x}_{j,i}$ iff $C_i \in \mathrm{F}(x_j)$,

(b) The set of edges is

$$\begin{array}{rcl} E^\Phi & = & \{(x_{n+1}, x_{n+1})\} \cup \{(x_{j,\max(\mathrm{T}(x_j))}, x_{j+1}), (\overline{x}_{j,\max(\mathrm{F}(x_j))}, x_{j+1}) \mid 1 \le j \le n\} \\ & \cup & \{(x_j, x_{j,\min(\mathrm{T}(x_j))}), (x_j, \overline{x}_{j,\min(\mathrm{F}(x_j))}) \mid 1 \le j \le n\} \\ & \cup & \{(x_{j,i}, x_{j,\mathsf{next}(i,\mathrm{T}(x_j))} \mid 1 \le j \le n, i < \max(\mathrm{T}(x_j))\} \\ & \cup & \{(\overline{x}_{j,i}, \overline{x}_{j,\mathrm{F}(i,\mathrm{T}(x_j))} \mid 1 \le j \le n, i < \max(\mathrm{F}(x_j))\}. \end{array}$$

We now explain the role if each set of edges. The first edge is the self-loop at x_{n+1}. The second set of edges specifies that from $x_{j,\max(\mathrm{T}(x_j))}$ the next vertex is x_{j+1} and similarly, from $\overline{x}_{j,\max(\mathrm{F}(x_j))}$ the next vertex is again x_{j+1}. The third set of edges specifies that from x_j there are two successors that are $x_{j,i}$ and $\overline{x}_{j,i'}$ where $i = \min(\mathrm{T}(x_j))$ and $i' = \min(\mathrm{F}(x_j))$. The final sets of edges specifies (a) to move in a line from $x_{j,\min(\mathrm{T}(x_j))}$ to visit the clauses that are satisfied by setting x_j as true, and (b) to move in a line from $\overline{x}_{j,\mathrm{F}(\mathrm{T}(x_j))}$ to visit the clauses that are satisfied by setting x_j as false. Fig 1 gives a pictorial view of the reduction.

(c) The initial vertex is $v_{in}^\Phi = x_1$.

(d) $\mathsf{AP}^\Phi = \{C_1, C_2, \ldots, C_m, X\}$, i.e., there is a proposition C_i for each clause C_i and there is a proposition X for all variables;

(e) $\mathcal{L}^\Phi(x_j) = X$; i.e., every variable state is labeled by the proposition X; and we have $\mathcal{L}^\Phi(x_{j,i}) = C_i$ and $\mathcal{L}^\Phi(\overline{x}_{j,i}) = C_i$, i.e., each state $x_{j,i}$ and $\overline{x}_{j,i}$ is labeled by the corresponding clause that is indexes.

The number of states in $\mathcal{G}^\Phi$ is $O(n \cdot m)$, and the reduction is polynomial in Φ. In this graph the maximal number of propositions visited is exactly equal to the maximal number of satisfiable clauses plus 1 (since along with the propositions for clauses the proposition X for all variables is always visited). The proof of the above claim is as follows. Given a path ω in G^Φ we construct an assignment A for the variables as follows: if the choice at a vertex x_j is $x_{j,\min(\mathrm{T}(x_j))}$, then we set x_j as true in A, else we set x_j as false. Hence if a path in $\mathcal{G}^\Phi$ visits a set $P \subseteq \mathsf{AP}^\Phi$ of r propositions, then the assignment A satisfies $r-1$ clauses (namely, $P \setminus \{X\}$). Conversely, given an assignment A of the variables, we construct a path ω^A in $\mathcal{G}^\Phi$ as follows: if x_j is true in the assignment A, then the path ω^A chooses $x_{j,\min(\mathrm{T}(x_j))}$ at x_j, otherwise, it chooses $\overline{x}_{j,\min(\mathrm{F}(x_j))}$ at x_j. If A satisfies a set Q of $r-1$ clauses, then ω^A visits $r+1$ propositions (namely, the set $Q \cup \{X\}$ of propositions). Hence Φ is satisfiable iff the answer to the maximal coverage problem with input $\mathcal{G}^\Phi$ and $m+1$ is true.

The desired result follows. ∎

Hardness of approximation. We note that from the proof Theorem 1 it follows that the MAX-SAT problem (i.e., computing the maximal number of clauses satisfiable for a SAT formula) can be reduced to the problem of computing the exact number for the maximal coverage problem. From hardness of approximation of the MAX-SAT problem [4], it follows that the maximal coverage problem for labeled graphs is hard to approximate.

Theorem 2 *The maximal coverage problem for labeled graphs that are controllably recurrent can be decided in PTIME.*

Proof. To solve the maximal coverage problem for labeled graphs that are controllably recurrent, we compute the maximal strongly connected component C that v_{in} belongs to. Since the graph is controllably recurrent, all states that are reachable from v_{in} belong to C. Hence the answer to the maximal coverage problem is "Yes" iff $|\bigcup_{v \in C} \mathcal{L}(v)| \geq m$. The result follows. ∎

3.2 Game graphs

Theorem 3 *The maximal coverage problem for labeled game graphs is PSPACE-complete.*

Proof. The proof consists of two parts. We present them below.

1. *In PSPACE.* We argue that the maximal coverage problem for labeled game graph can be reduced to the coverage in bounded time problem. The reason is as follows: in a labeled game graph with n vertices, if player 1 can visit m propositions, then player 1 can visit m propositions within at most $m \cdot n$ steps; because player 1 can always play a strategy from the current position that visits a new proposition that is not visited and never needs to go through a cycle without visiting a new proposition unless the maximal coverage is achieved. Hence it follows that the maximal coverage problems for games reduces to the coverage in bounded time problem. The PSPACE inclusion will follow from the result of Theorem 7 where we show that the coverage in bounded time problem is in PSPACE.

2. *PSPACE-hardness.* The maximal coverage problem for game graphs is PSPACE-complete, even if the underlying graph is strongly connected. The proof is a reduction from QBF (truth of quantified boolean formulas) that is known to be PSPACE-complete [12], and it is a modification of the reduction of Theorem 1. Consider a QBF formula

$$\Phi = \exists x_1 . \forall x_2 . \exists x_3 \ldots \exists x_n . C_1 \wedge C_2 \wedge \ldots C_m;$$

defined on the set $X = \{x_1, x_2, \ldots, x_n\}$ of variables, and $C_1, C_2, \ldots, C_m$ are the clauses of the formula. We apply the reduction of Theorem 1 with the following modification to obtain the labeled game graph $\mathcal{G}^\Phi$: the partition (V_1^Φ, V_2^Φ) of V^Φ is as follows. For a variable x_j if the quantifier before x_j is existential, then $x_j \in V_1^\Phi$ (i.e., for existentially quantified variable, player 1 chooses the out-going edges denoting whether to set the variable true or false); and for a variable x_j if the quantifier before x_j is universal, then $x_j \in V_2^\Phi$ (i.e., for universally quantified variable, the opposing player 2 chooses the out-going edges denoting whether to set the variable true or false). The state x_{n+1} is a player 2 vertex, and all other vertex has an single out-going edges and can be player 1 state. Given this game graph we have Φ is true iff player 1 can ensure that all the propositions can be visited in $\mathcal{G}^\Phi$. Formally, let Π_1^Φ and Π_2^Φ denote the set of all strategies for player 1 and player 2, respectively, in $\mathcal{G}^\Phi$. Then Φ is true iff $\sup_{\pi_1 \in \Pi_1^\Phi} \inf_{\pi_2 \in \Pi_2^\Phi} |\mathcal{L}^\Phi(\omega(x_1, \pi_1, \pi_2))| \geq m + 1$. Observe that since x_{n+1} is a player 2 state if we add an edge from x_{n+1} to x_1, player 2 will never choose the edge x_{n+1} to x_1 (since the objective for player 2 is to minimize the coverage). However, adding the edge from x_{n+1} to x_1 makes the underlying graph strongly connected (i.e., the underlying graph of the game graph becomes controllably recurrent; but player 1 does not have a strategy to ensure that x_1 is reached, so the game is not controllably recurrent).

The desired result follows. ∎

Complexity of maximal coverage in controllably recurrent games. We will now consider maximal coverage in controllably recurrent games. Our analysis will use fixing memoryless *randomized* strategy for player 1, and fixing a memoryless randomized strategy in labeled game graph we get a labeled Markov decision process (MDP). A labeled MDP consists of the same components as a labeled game graph, and for vertices in V_1 (which are randomized vertices in the MDP) the successors are chosen uniformly at random (i.e., player 1 does not have a proper choice of the successor but chooses all of them uniformly at random). Given a labeled game graph $\mathcal{G} = ((V, E), (V_1, V_2), v_{in}, \mathsf{AP}, \mathcal{L})$ we denote by $\mathsf{Unif}(\mathcal{G})$ the MDP interpretation of $\mathcal{G}$ where player 1 vertices chooses all successors uniformly at random. An *end component* in $\mathsf{Unif}(G)$ is a set U of vertices such that (i) U is strongly connected and (ii) U is player 1 *closed*, i.e., for all $u \in U \cap V_1$, for all u' such that $(u, u') \in E$ we have $u' \in U$ (in other words, for all player 1 vertices, all the out-going edges are contained in U).

Lemma 1 *Let $\mathcal{G}$ be a labeled game graph and let $\mathsf{Unif}(\mathcal{G})$ be the MDP interpretation of $\mathcal{G}$. Then the following assertions hold.*

1. *Let U be an end-component in $\mathsf{Unif}(\mathcal{G})$ with $v_{in} \in U$. Then $\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2))| \leq |\bigcup_{u \in U} \mathcal{L}(u)|$.*
2. *There exists an end-component $U \in \mathsf{Unif}(\mathcal{G})$ with $v_{in} \in U$ such that $|\bigcup_{u \in U} \mathcal{L}(u)| \leq \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2))|$.*

Proof. We prove both the claims below.

1. If U is an end-component in $\mathsf{Unif}(G)$, then consider a memoryless strategy π_2^* for player 2, that for all vertices $u \in U \cap V_2$, chooses a successor $u' \in U$ (such a successor exists since U is strongly connected). Since U is player 1 closed (i.e., for all player 1 out-going edges from U, the end-point is in U), it follows that for all strategies of player 1 , given the strategy π_2^* for player 2, the vertices visited in a play is contained in U. The desired result follows.
2. An optimal strategy π_1^* for player 1 in $\mathcal{G}$ is as follows:
   (a) Let $Z_0 = \{v \in V \mid \mathcal{L}(v) = \mathcal{L}(v_{in})\}$ and $i = 0$;
   (b) At iteration i, let Z_i represents the set of propositions already visited. At iteration i, player 1 plays a strategy to reach a state in $V \setminus Z_i$ (if such a strategy exists), and then reaches back v_{in} (a strategy to reach back v_{in} always exists since the game is controllably recurrent).
   (c) If a new proposition p_i is visited at iteration i, then let $Z_{i+1} = Z_i \cup \{v \in V \mid \mathcal{L}(v) = p_i\}$. Goto step (b) for $i + 1$ iteration with Z_{i+1}. If no state in $V \setminus Z_i$ can be reached, then stop.

   The strategy π_1^* is optimal, and let the above iteration stop with $Z_i = Z^*$. Let $X = V \setminus Z^*$, and let X^* be the set of vertices such that player 1 can reach X. Let $U^* = V \setminus X^*$. Then $v_{in} \in U^*$ and player 2 can ensure that from v_{in} the game can be confined to U^*. Hence the following conditions must hold: (a) for all $u \in U^* \cap V_2$, there exists $u' \in U^*$ such that

$(u, u') \in E$; and (b) for all $u \in U^* \cap V_1$, for all $u' \in V$ such that $(u, u') \in E$ we have $u' \in U^*$. Consider the sub-graph G' where player 2 restricts itself to edges only in U^*. A bottom maximal strongly connected component $U \subseteq U^*$ in the sub-graph is an end-component in $\mathsf{Unif}(\mathcal{G})$, and we have

$$|\bigcup_{u \in U} \mathcal{L}(u)| \leq |\bigcup_{u \in U^*} \mathcal{L}(u)| \leq |\bigcup_{u \in Z^*} \mathcal{L}(u)|.$$

It follows that U is a witness end-component to prove the result.

The desired result follows. ∎

Theorem 4 *The maximal coverage problem for labeled game graphs that are controallably recurrent is coNP-complete.*

Proof. We prove the following two claims to establish the result.

1. *In coNP.* The fact that the problem is in coNP can be proved using Lemma 1. Given a labeled game graph $\mathcal{G}$, if the answer to the maximal coverage problem (i.e., whether $\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} |\mathcal{L}(\omega(v_{in}, \pi_1, \pi_2))| \geq m$) is NO, then by Lemma 1, there exists an end-component U in $\mathsf{Unif}(\mathcal{G})$ such that $|\bigcup_{u \in U} \mathcal{L}(u)| < m$. The witness end-component U is a polynomial witness and it can be guessed and verified in polynomial time. The verification that U is the correct witness is as follows: we check (a) U is strongly connected; (b) for all $u \in U \cap V_1$ and for all $u' \in V$ such that $(u, u') \in E$ we have $u' \in U$; and (c) $|\bigcup_{u \in U} \mathcal{L}(u)| < m$. Hence the result follows.

2. *coNP hardness.* We prove hardness using a reduction from the complement of the *Vertex Cover* problem. Given a graph $G = (V, E)$, a set $U \subseteq V$ is a *vertex cover* if for all edges $e = (u_1, u_2) \in E$ we have either $u_1 \in U$ or $u_2 \in U$. Given a graph G whether there is a vertex cover U of size at most m (i.e., $|U| \leq m$) is NP-complete [8]. We now present a reduction of the complement of the vertex cover problem to the maximal coverage problem in controallably recurrent games. Given a graph $G = (V, E)$ we construct a labeled game graph $\overline{\mathcal{G}}$ as follows. Let the set E of edges be enumerated as $\{e_1, e_2, \ldots, e_\ell\}$, i.e., there are ℓ edges. The labeled game graph $\overline{\mathcal{G}} = ((\overline{V}, \overline{E}), (\overline{V}_1, \overline{V}_2), v_{in}, \mathsf{AP}, \mathcal{L})$ is as follows.

   (a) *Vertex set and partition.* The vertex set $\overline{V}$ is as follows:

   $$\overline{V} = \{v_{in}\} \cup E \cup \{e_i^j \mid 1 \leq i \leq \ell, 1 \leq j \leq 2\}.$$

   All states in E are player 2 states, and the other states are player 1 states, i.e., $\overline{V}_2 = E$, and $\overline{V}_1 = \overline{V} \setminus \overline{V}_2$.

   (b) *Edges.* The set $\overline{E}$ of edges are as follows:

   $$\begin{aligned} \overline{E} \;=\; & \{(v_{in}, e_j) \mid 1 \leq j \leq \ell\} \;\cup\; \{(e_i, e_i^j) \mid 1 \leq i \leq \ell, 1 \leq j \leq 2\} \\ & \cup \{(e_i^j, v_{in}) \mid 1 \leq i \leq \ell, 1 \leq j \leq 2\}. \end{aligned}$$

   Intuitively, the edges in the game graph are as follows: from the initial vertex v_{in}, player 1 can choose any of the edges $e_i \in E$. For a vertex e_i in $\overline{V}$, player 2 can choose between two vertices e_i^1 and e_i^2 (which will eventually represent the two end-points of the edge e_i). From vertices of the form e_i^1 and e_i^2, for $1 \leq i \leq \ell$, the next vertex is the initial vertex v_{in}. It follows that from all vertex the game always comes back to v_{in} and hence we have controllably recurrent game.

(c) *Propositions and labelling.* $\mathsf{AP} = V \cup \{\$ \mid \$ \notin V\}$, i.e., there is a proposition for every vertex in V and a special proposition \$. The vertex v_{in} and vertices in E are labeled by the special proposition \$, i.e., $\mathcal{L}(v_{in}) = \$$; and for all $e_i \in E$ we have $\mathcal{L}(e_i) = \$$. For a vertex e_i^j, let $e_i = (u_i^1, u_i^2)$, where u_i^1, u_i^2 are vertices in V, then $\mathcal{L}(e_i^1) = u_i^1$ and $\mathcal{L}(e_i^2) = u_i^2$. Note that the above proposition assignment ensures that at every vertex that represents an edge, player 2 has the choices of vertices that form the end-points of the edge.

The following case analysis completes the proof.

- Given a vertex cover U, consider a player 2 strategy, that at a vertex $e_i \in \overline{V}$, choose a successor e_i^j such that $\mathcal{L}(e_i^j) \in U$. The strategy for player 2 ensures that player 1 visits only propositions in $U \cup \{\$\}$, i.e., at most $|U| + 1$ propositions.
- Consider a strategy for player 1 that from v_{in} visits all states $e_1, e_2, \ldots, e_\ell$ in order. Consider any counter-strategy for player 2 and let $U \subseteq V$ be the set of propositions other than \$ visited. Since all the edges are chosen, it follows that U is a vertex cover. Hence if all vertex cover in G is of size at least m, then player 1 can visit at least $m+1$ propositions.

Hence there is a vertex cover in G of size at most m if and only if the answer to the maximal coverage problem in $\mathcal{G}$ with $m+1$ is NO. It follows that the maximal coverage problem in controllably recurrent games is coNP-hard.

The desired result follows. ∎

Complexity of minimal safety games. As a corollary of the proof of Theorem 4 we obtain a complexity result about *minimal safety games*. Given a labeled game graph $\mathcal{G}$ and m, the minimal safety game problem asks, whether there exists a set U such that a player can confine the game in U and U contains at most m propositions. An easy consequence of the hardness proof of Theorem 4 is minimal safety games are NP-hard, and also it is easy to argue that minimal safety games are in NP. Hence we obtain that the minimal safety game problem is NP-complete.

4 The Complexity of Coverage in Bounded Time Problem

In this section we study the complexity of the coverage in bounded time problem. In subsection 4.1 we study the complexity for graphs, and in subsection 4.2 we study the complexity for game graphs.

4.1 Graphs

Theorem 5 *The coverage in bounded time problem for both labeled graphs and controllably recurrent labeled graphs is NP-complete.*

Proof. We prove the completeness result in two parts below.

1. *In NP.* Given a labeled graph with n vertices, if there a path ω such that $|\mathcal{L}(\omega \restriction k)| \geq m$, then there is path ω' such that $|\mathcal{L}(\omega' \restriction m \cdot n)| \geq m$. The above claim follows since any cycle that does not visit any new proposition can be omitted. Hence a path of length $j = \min(k, m \cdot n)$ can be guessed and it can be then checked in polynomial time if the path of length j visits at least m propositions.

2. *In NP-hard.* We reduce the *Hamiltonian-path (HAM-PATH)* [8] problem to the coverage in bounded time problem for labeled graphs. Given a directed graph $G = (V, E)$ and an initial vertex v, we consider the labeled graph $\mathcal{G}$ with the directed graph G, with v as the initial state and $\mathsf{AP} = V$ and $\mathcal{L}(u) = u$ for all $u \in V$, i.e., each vertex is labeled with an unique proposition. The answer to the coverage is bounded time with $k = n$ and $m = n$, for $n = |V|$ is "YES" iff there is a HAM-PATH in G starting from v.

The desired result follows. ∎

Complexity in size of the graph. We now argue that the maximal coverage and the coverage in bounded time problem on labeled graphs can be solved in non-deterministic log-space in the size of the graph, and polynomial space in the size of the atomic propositions. Given a labeled graph $\mathcal{G}$, with n vertices, we argued in Theorem 1 that if m propositions can be visited, then there is a path of length at most $m \cdot n$, that visits m propositions. The path of length $m \cdot n$, can be visited, storing the current vertex, and guessing the next vertex, can checking the set of propositions already visited. Hence this can be achieved in non-deterministic log-space in the size of the graph, and polynomial space in the size of the proposition set. A similar argument holds for the coverage in bounded time problem. This gives us the following result.

Theorem 6 *Given a labeled graph $\mathcal{G} = ((V, E), v_{in}, \mathsf{AP}, \mathcal{L})$, the maximal coverage problem and the coverage in bounded time problem can be decided in NLOGSPACE in $|V| + |E|$, and in PSPACE in $|\mathsf{AP}|$.*

4.2 Game graphs

Theorem 7 *The coverage in bounded time problem for labeled game graphs is PSPACE-complete.*

Proof. We prove the following two cases to prove the result.

1. *PSPACE-hardness.* It follows from the proof of Theorem 3 that the maximal coverage problem for labeled game graphs reduces to the coverage in bounded time problem for labeled game graphs. Since the maximal coverage problem for labeled game graphs is PSPACE-hard (Theorem 3), the result follows.

2. *In PSPACE.* We say that an *exploration game tree* for a labeled game graph is a rooted, labeled tree which represents an unfolding of the graph. Every node α of the tree is labeled with a pair (v, b), where v is a node of the game graph, and $b \subseteq \mathsf{AP}$ is the set of propositions that have been visited in a branch leading from the root of the tree to α. The root of the tree is labeled with $(v_{in}, \mathcal{L}(v_{in}))$. A tree with label (v, b) has one descendant for each u with $(v, u) \in E$; the label of the descendant is $(u, b \cup \mathcal{L}(u))$.

   In order to check if m different propositions can be visited within k-steps, the PSPACE algorithm traverses the game tree in depth first order. Each branch is explored up to one of the two following conditions is met: (i) depth k is reached, or (ii) a node is reached, which has the same label as an ancestor in the tree. The bottom nodes, where conditions (i) or (ii) are met, are thus the leaves of the tree. In the course of the traversal, the algorithm computes in bottom-up fashion the *value* of the tree nodes. The value of a leaf node labeled (v, b) is $|b|$. For player-1 nodes, the value is the maximum of the values of the successors; for player-2

nodes, the value is the minimum of the value of the successors. Thus, the value of a tree node α represents the minimum number of propositions that player 1 can ensure are visited, in the course of a play of the game that has followed a path from the root of the tree to α, and that can last at most k steps. The algorithm returns Yes if the value at the root is at least m, and no otherwise.

To obtain the PSPACE bound, notice that if a node with label (v, b) is an ancestor of a node with label (v', b') in the tree, we have $b \subseteq b'$: thus, along a branch, the set of propositions appearing in the labels increases monotonically. Between two increases, there can be at most $|\mathcal{G}|$ nodes, due to the termination condition (ii). Thus, each branch needs to be traversed at most to depth $1 + |\mathcal{G}| \cdot (|\mathsf{AP}| + 1)$, and the process requires only polynomial space.

The result follows. ∎

Theorem 8 *The coverage in bounded time problem for labeled game graphs that are controllably recurrent is both NP-hard and coNP-hard, and can be decided in PSPACE.*

Proof. It follows from the (PSPACE-inclusion) argument of Theorem 3 that the maximal coverage problem for labeled game graphs that are controllably recurrent can be reduced to the coverage in bounded time problem for labeled game graphs that are controllably recurrent. Hence the coNP-hardness follows from Theorem 4, and the NP-hardness follows from hardness in labeled graphs that are controllably recurrent (Theorem 5). The PSPACE-inclusion follows from the general case of labeled game graphs (Theorem 7). ∎

Theorem 8 shows that for controllably recurrent game graphs, the coverage in bounded time problem is both NP-hard and coNP-hard, and can be decided in PSPACE. A tight complexity bound remains an open problem.

Complexity in the size of the game. The maximal coverage problem can alternately be solved in time linear in the size of the game graph and exponential in the number of propositions. Given a game graph $G = ((V, E), (V_1, V_2), v_{in}, \mathsf{AP}, \mathcal{L})$, construct the game graph $G' = ((V', E'), (V_1', V_2'), v_{in}', \mathsf{AP}, \mathcal{L}')$ where $V' = V \times 2^{\mathsf{AP}}$, $((v, b), (v', b')) \in E'$ iff $(v, v') \in E$ and $b' = b \cup \mathcal{L}(v')$, $V_i = \{(v, b) \mid v \in V_i\}$ for $i \in \{1, 2\}$, $v_{in}' = (v_{in}, \mathcal{L}(v_{in}))$, and $\mathcal{L}'(v, b) = \mathcal{L}'(v)$. Clearly, the size of the game graph G' is linear in G and exponential in AP. Now consider a reachability game on G' with the goal $\{(v, b) \mid v \in V \text{ and } |b| \geq m\}$. Player-1 wins this game iff the maximal coverage problem is true for G and m propositions. Since a reachability game can be solved in time linear in the game, the result follows. A similar construction, where we additionally track the length of the game so far, shows that the maximal coverage problem with bounded time can be solved in time linear in the size of the game graph and exponential in the number of propositions.

Theorem 9 *Given a labeled game graph $\mathcal{G} = ((V, E), (V_1, V_2), v_{in}, \mathsf{AP}, \mathcal{L})$ the maximal coverage and the coverage in bounded time problem can be solved in linear-time in $O(|V| + |E|)$ and in exponential time in $|\mathsf{AP}|$.*

5 Extensions

Somewhat surprisingly, despite the central importance of graph coverage in system verification, several basic complexity questions have remained open. The basic setting of this paper on graphs and games can be extended in various directions, enabling the modeling of other system features. We mention two such directions.

Incomplete Information. So far, we have assumed that at each step, the tester has complete information about the state of the system under test. In practice, this may not be true, and the tester might be able to observe only a part of the state. This leads to graphs and games of *imperfect information* [13]. The maximal coverage and the coverage in bounded time problem for games of imperfect information can be solved in EXPTIME. The algorithm first constructs a perfect-information game graph by subset construction [13], and then run the algorithm of Theorem 9, that is linear in the size game graph and exponential in the number of propositions, on the perfect-information game graph. Thus, the complexity of this algorithm is EXPTIME. The reachability problem for imperfect-information games is already EXPTIME-hard [13], hence we obtain an optimal EXPTIME-complete complexity.

Timed Systems. Second, while we have studied the problem in the discrete, finite-state setting, similar questions can be studied for timed systems modeled as timed automata [2] or timed game graphs [11]. Such problems would arise in the testing of real-time systems. We omit the standard definitions of timed automata and timed games. The maximal coverage problem for timed automata (respectively, timed games) takes as input a timed automaton T (respectively, a timed game T), with the locations labeled by a set AP of propositions, and a number m, and asks whether m different propositions can be visited. An algorithm for the maximal coverage problem for timed automata constructs the region graph of the automaton [2] and runs the algorithm of Theorem 6 on the labeled region graph. This gives us a PSPACE algorithm. Since the reachability problem for timed automata is PSPACE-hard, we obtain a PSPACE-complete complexity. Similar result holds for the coverage in bounded time problem for timed automata. Similarly, the maximal coverage and coverage in bounded time problem for timed games can be solved in exponential time by running the algorithm of Theorem 9 on the region game graph. This gives an exponential time algorithm. Again, since game reachability on timed games is EXPTIME-hard [9], we obtain that maximal coverage and coverage in bounded time in timed games is EXPTIME-complete.

References

[1] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *Proc. 27th ACM Symp. Theory of Comp.*, 1995.

[2] R. Alur and D. Dill. The theory of timed automata. In *Real-Time: Theory in Practice*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 45–73. Springer-Verlag, 1991.

[3] P. Ammann and J. Offutt. *Introduction to software testing.* Cambridge University Press, 2008.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[5] A. Blass, Y. Gurevich, L. Nachmanson, and M. Veanes. Play to test. In *FATES*, volume 3997 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2005.

[6] E. Brinksma and J. Tretmans. Testing transition systems: An annotated bibliography. In *MOVEP 00: Modeling and Verification of Parallel Processes*, volume 2067 of *Lecture Notes in Computer Science*, pages 187–195. Springer, 2000.

[7] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Math. Prog.*, 5:88–124, 1973.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co., 1979.

[9] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[10] D. Lee and M. Yannakakis. Optimization problems from feature testing of communication protocols. In *ICNP: International Conference on Network Protocols*, pages 66–75. IEEE Computer Society, 1996.

[11] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. of 12th Annual Symp. on Theor. Asp. of Comp. Sci.*, volume 900 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1995.

[12] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1993.

[13] J.H. Reif. The compexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29:274–301, 1984.

[14] M. Yannakakis. Testing, optimizaton, and games. In *LICS*, pages 78–88. IEEE Computer Society, 2004.

[15] M. Yannakakis and D. Lee. Testing for finite state systems. In *CSL 98: Computer Science Logic*, volume 1584 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 1998.